EECS 1012: LAB 07 – more on JavaScript; intro to unit-testing

A. IMPORTANT REMINDERS

1) You must attend your assigned lab session (we will be marking your submission in the lab).
2) You are required to arrive on time: anyone later than 15 minutes may not be admitted to the lab.
3) You are required to complete the pre-lab mini quiz posted on Moodle in the first 15 minutes of your lab time.
4) Each lab including the pre-lab mini quiz is about 1.5% of your overall grade.
5) TAs are in the lab to help you. They will also verify and mark your work at the end. Signal a TA for help if you stuck on any of the steps below. Yet, note that TAs would need to help other students too.

B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

1) Complete your My Learning Kit Project (that you started from Lab03) with 20 problem definitions, flowcharts, and JavaScript Solution. Don't need to do any task for the "another solution" panel yet. At the beginning of Lab07 this week, as soon as you are done with the pre-lab quiz, show your My Learning Kit project to your TA for credit. TAs are instructed NOT to accept this project after 20 minutes of the lab official time. Hence, you should make sure your project is ready before you go to the lab. As you will need this project for Lab08 too, failing to complete it by Lab07 may affect your grade in Lab08 too.
2) Download this lab files and read them carefully to the end.
3) You should have a good understanding of
   - JavaScript objects, such as **Math**, **Date**, and DOM **document**
   - **array** and **string** in JavaScript
4) `Also, see assert API here:` https://www.chaijs.com/api/assert/

C. GOALS/OUTCOMES FOR LAB

1) To practice more concepts in programming, including variables, arrays, functions (aka sub-algorithms), and program control statements.
2) To use more JS objects, such as `document`, `Math`, and `Date`.
3) To become familiar with unit-testing.

D. TASKS

Part 0: For unit-testing in JavaScript, you may want to install `Mocha` and `Chai` on your own computer. For that,

1) Install **node js**
2) Navigate to your folder and run: **npm init** (initialize app and create a package.json file)
3) Install `Mocha` and `Chai`: **npm install mocha chai --save-dev**
4) In **package.json** file replace test line with

```
"scripts": {
    "test": "mocha || true"
  }
```

Part 1:

1) TASK 1: Simple "HEADS" or "TAILS" button output with an if-statement. You also run some already prepared unit testing code for this task.
2) TASK 2: Passing variables to functions. You also run some already prepared unit testing code for this task.
3) TASK 3: Passing variables and for-loop. You also run some already prepared unit testing code for this task.
4) TASK 4: Random + string concatenation + if-statement. You write some unit testing code for this task and then run it.
5) TASK 5: Date object + array + string concatenation.
6) TASK 6: Global variable and if-statement

E. SUBMISSIONS

1) Manual verification by a TA

You will need to have one of the TAs of your lab to verify your work before submission. The TA will look at your various files in their progression. The TA may ask you to make minor modifications to the lab to demonstrate your knowledge of the materials. The TA will mark your name off a list; You are required to sign the list to show your attendance and that you have been verified.

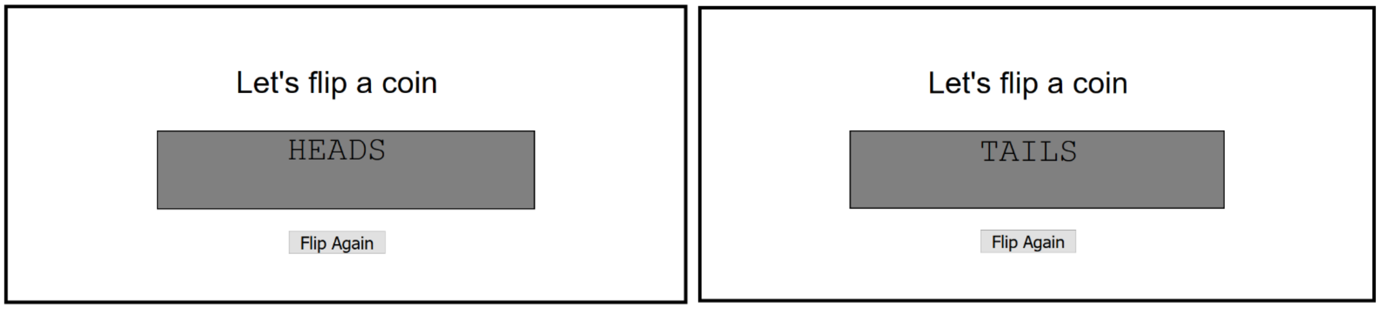
2) Moodle submission

Create a **folder** named "**Lab07**" and copy **all** of your HTML and JS files, including the tests and the testRunner. Once you are done, compress the folder and upload the zip (or tar) file to moodle. Also, create a folder name "**LearningKit**" and copy all of your Learning Kit materials; then, compress it and upload the zip/tar file to moodle.

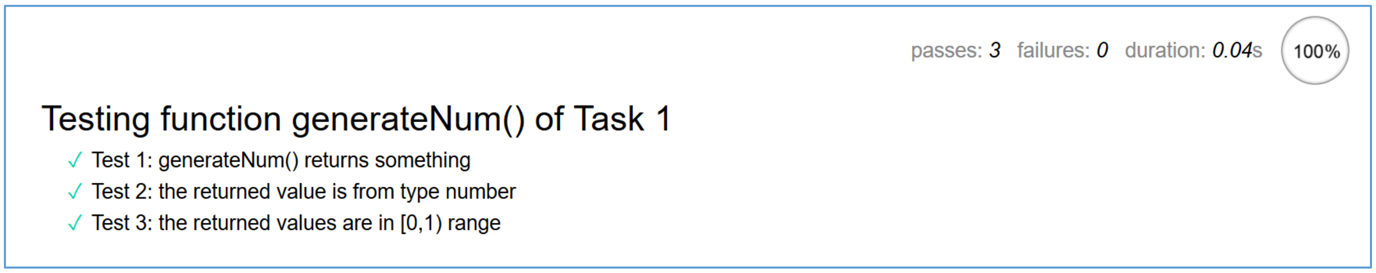
F. FURTHER DETAILS

Task 1: Edit task1.js (you do not need to edit the HTML file).

For this task, we have already declared the JavaScript function myFunction() for you. Your function should do the following. Each time the button is clicked, your myFunction() code should call a sub-algorithm that generates a random number between 0 and 1 (including 0 and excluding 1). If the random number is less than 0.5, then have the innerHTML of the paragraph variable set to "HEADS", otherwise set it to "TAILS". See example outputs below.



Once you are done, double click on `testRunner1.html`. If you have implemented task1.js properly, you should see the following information in your web browser, meaning that all 3 tests have passed. If any test fails, you must go back to your task1.js and debug your code.



passes: 3 failures: 0 duration: 0.04s 100%

Testing function generateNum() of Task 1

✓ Test 1: generateNum() returns something

✓ Test 2: the returned value is from type number

✓ Test 3: the returned values are in [0,1) range

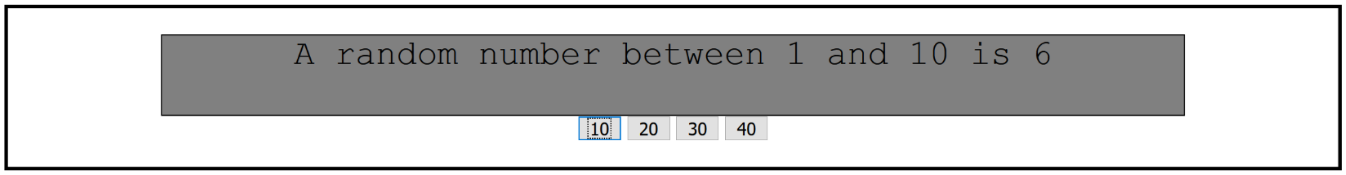
Task 2. Edit task2.html and task2.js

(1) Link your task2.js to your HTML code.

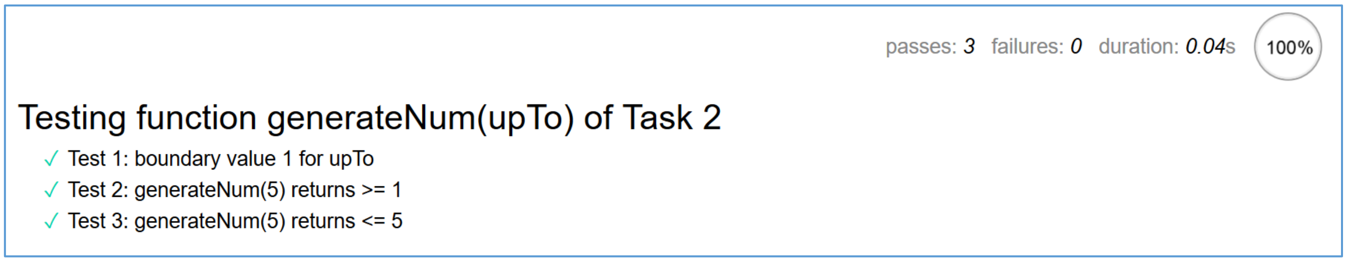
(2) Have the text in the paragraph "mydata" start with **Result** (see below).

(2) Add four buttons to your `Task2.html` as shown below.

(3) Write a function in JavaScript that has one parameter. When a button is pressed, it should pass the value shown in the button (e.g., 10, 20, 30, or 40) to a function named `passNum`. In your JavaScript code, your function should call a sub-algorithm that generates a random whole number between 1 and the passed value, inclusively. See example below for when the button that passes value 10 is pressed.



Once you are done, double click on `testRunner2.html`. If you have implemented task2.js properly, you should see the following information in your web browser, meaning that all 3 tests have passed. If any test fails, you must go back to your task2.js and debug your code.

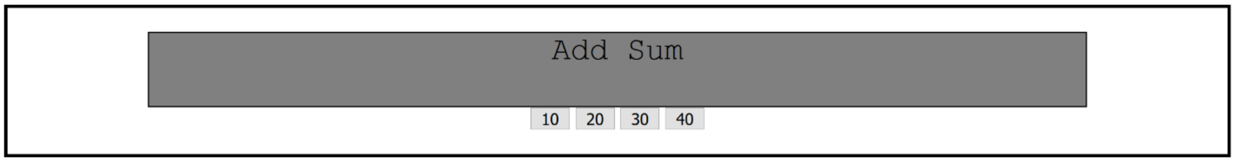


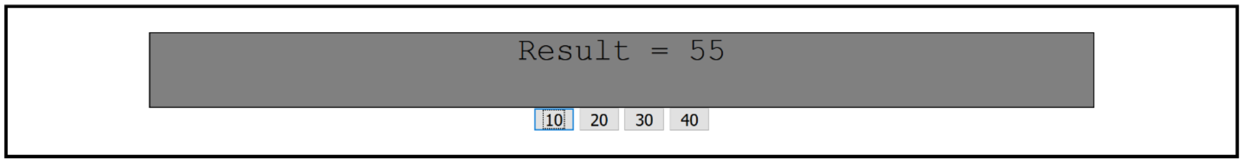
```
passes: 3   failures: 0   duration: 0.04s   100%

Testing function generateNum(upTo) of Task 2
  ✓ Test 1: boundary value 1 for upTo
  ✓ Test 2: generateNum(5) returns >= 1
  ✓ Test 3: generateNum(5) returns <= 5
```

Task 3. Edit task3.js (you do not need to edit the HTML file).

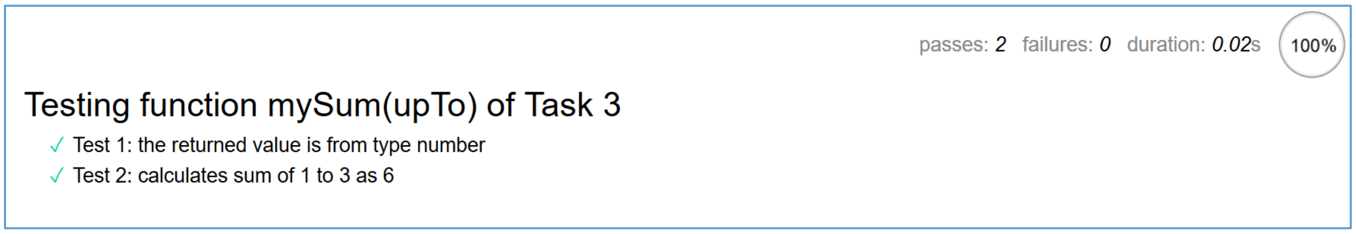
Write a function in JavaScript that has one parameter. When each button is pressed, it should pass the *value* shown in the button (e.g., 10, 20, 30, or 40). Use a for-loop to compute the sum of 0 to the passed *value*.

For example, if the value passed is 10, then compute 0+1+2+3+4+5+6+7+8+9+10=55. See below.





Once you are done, double click on `testRunner3.html`. If you have implemented task3.js properly, you should see the following information in your web browser, meaning that both 2 tests have passed. If any test fails, you must go back to your task3.js and debug your code.



```
passes: 2   failures: 0   duration: 0.02s   100%

Testing function mySum(upTo) of Task 3
  ✓ Test 1: the returned value is from type number
  ✓ Test 2: calculates sum of 1 to 3 as 6
```

Task 4. Modify task4.html and task4.js

(1) Link your JavaScrpit file to your HTML file.

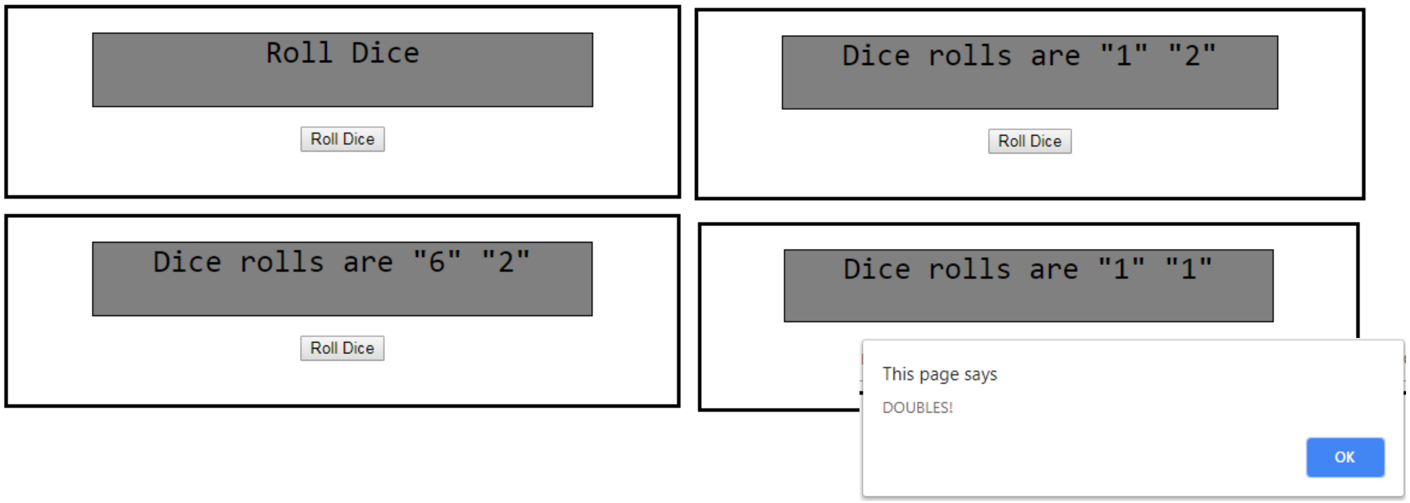
(2) Have the text in the paragraph "mydata" start with **Roll Dice**. Add a button "Roll Dice". Have this button respond the click event.

(3) Have the `onlick` for your button link to your JavaScript function. The function does not have parameters.

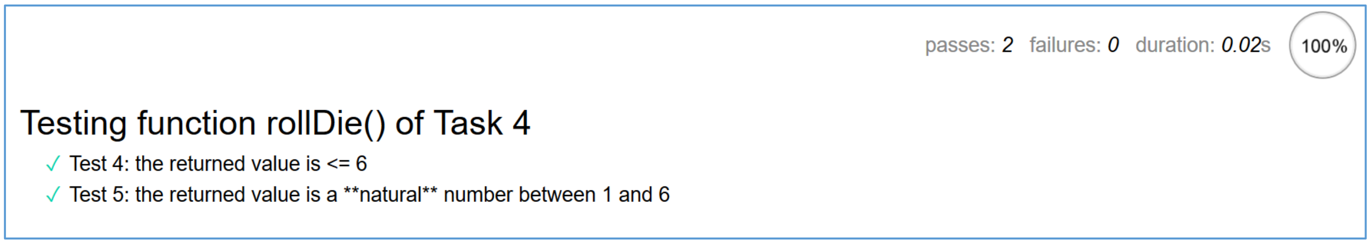
(4) Each time you click, have your JavaScript function compute two random numbers from 1 to 6. These represent dice. Change the `innerHTML` to say Dice rolls are “value1” and “value2”, where value1 and value2 are the results of your random numbers.

(5) If the two numbers are the equal, create an alert that says “DOUBLES!”.

See examples below.

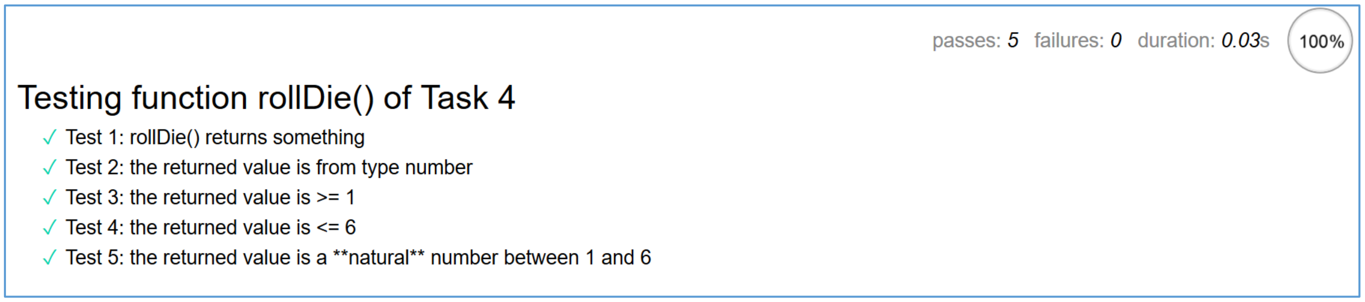


Once you have completed the steps above, double click on `testRunner4.html`. You should see the following figure. Otherwise, you must go to your task4.js and debug your code.



```
passes: 2   failures: 0   duration: 0.02s   100%

Testing function rollDie() of Task 4
  ✓ Test 4: the returned value is <= 6
  ✓ Test 5: the returned value is a **natural** number between 1 and 6
```

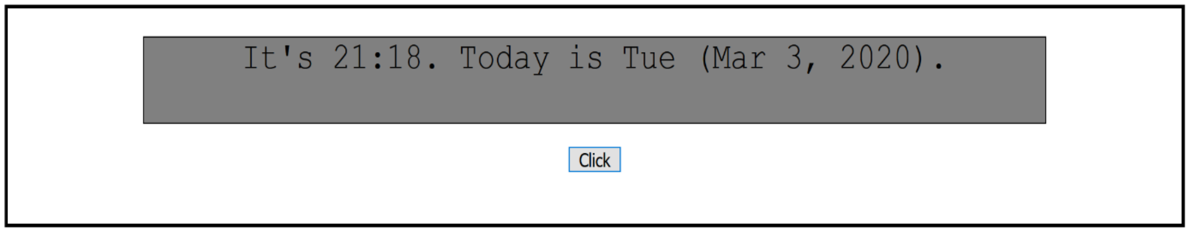
But, as you see in the figure above, we only see the results of Tests 4 and 5. In other words, Tests 1 to 3 are missing, and your task is to add them to `test/task4rollDicTest.js`. Open `task4rollDicTest.js` and follow the instructions there in TODO 1 to TODO 3. Once you are done, you should see the following figure.



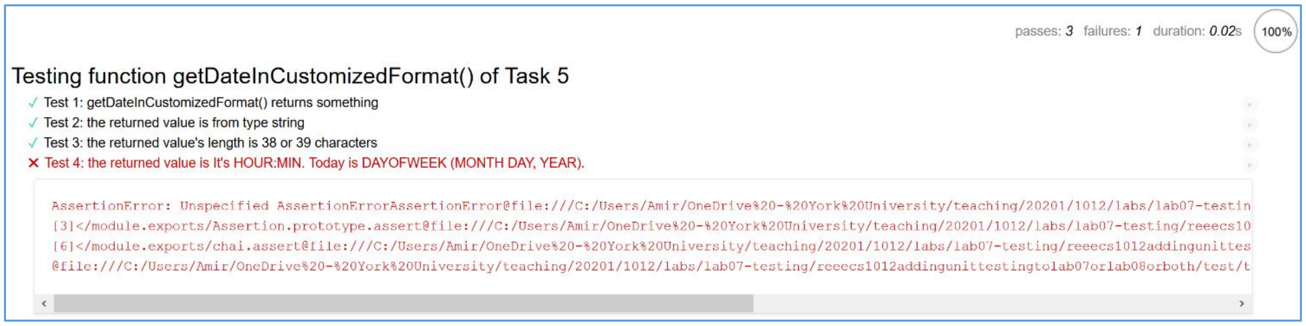
```
passes: 5   failures: 0   duration: 0.03s   100%

Testing function rollDie() of Task 4
  ✓ Test 1: rollDie() returns something
  ✓ Test 2: the returned value is from type number
  ✓ Test 3: the returned value is >= 1
  ✓ Test 4: the returned value is <= 6
  ✓ Test 5: the returned value is a **natural** number between 1 and 6
```

Task 5. Modify task5.js (you do not need to edit the HTML file).

In `task5.js`, complete line 27 such that when `task5.html` runs, the current time and date are shown with a format similar to the figures below.


It's 21:18. Today is Tue (Mar 3, 2020).

Once you have completed the step above, double click on `testRunner5.html`. You should see the following figure. If any of the first 3 tests have failed, you must go to your `task5.js` and debug your code.

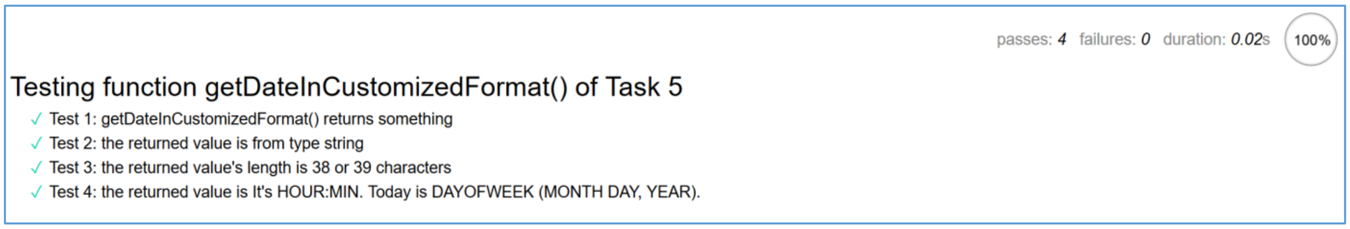

passes: 3 failures: 1 duration: 0.02s 100%

Testing function getDateInCustomizedFormat() of Task 5

✓ Test 1: getDateInCustomizedFormat() returns something
✓ Test 2: the returned value is from type string
✓ Test 3: the returned value's length is 38 or 39 characters
✕ Test 4: the returned value is It's HOUR:MIN. Today is DAYOFWEEK (MONTH DAY, YEAR).

```
AssertionError: Unspecified AssertionErrorAssertionError@file:///C:/Users/Amir/OneDrive%20-%20York%20University/teaching/20201/1012/labs/lab07-testin
[3]</module.exports/Assertion.prototype.assert@file:///C:/Users/Amir/OneDrive%20-%20York%20University/teaching/20201/1012/labs/lab07-testing/reeecs10
[6]</module.exports/chai.assert@file:///C:/Users/Amir/OneDrive%20-%20York%20University/teaching/20201/1012/labs/lab07-testing/reeecs1012addingunittes
@file:///C:/Users/Amir/OneDrive%20-%20York%20University/teaching/20201/1012/labs/lab07-testing/reeecs1012addingunittestingtolab07orlab08orboth/test/t
```

In order to debug the error in Test 4, open `test/task5getDateInCustomizedFormatTest.js`; go to the lines related to Test 4, replace the message with the exact current time and date of your computer, and run the test immediately. You should see the following result.


passes: 4 failures: 0 duration: 0.02s 100%

Testing function getDateInCustomizedFormat() of Task 5

✓ Test 1: getDateInCustomizedFormat() returns something
✓ Test 2: the returned value is from type string
✓ Test 3: the returned value's length is 38 or 39 characters
✓ Test 4: the returned value is It's HOUR:MIN. Today is DAYOFWEEK (MONTH DAY, YEAR).

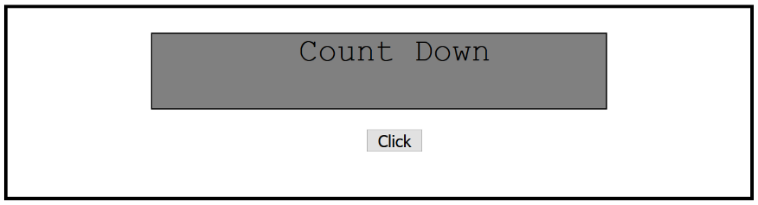
Task 6. Modify task6.js (you do not need to edit the HTML file).

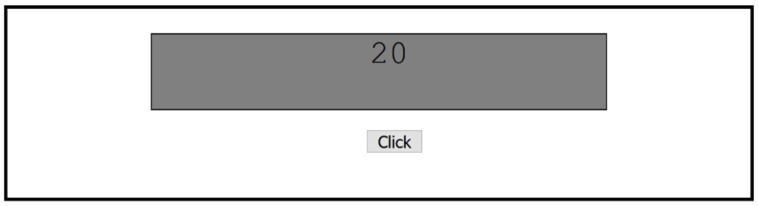
In task6.js, declare a global variable. This is a variable that is created outside your function. Inside your function, you do not need to declare it again. If you modify the variable, the modification will be remembered next time you access the function. See example code here.

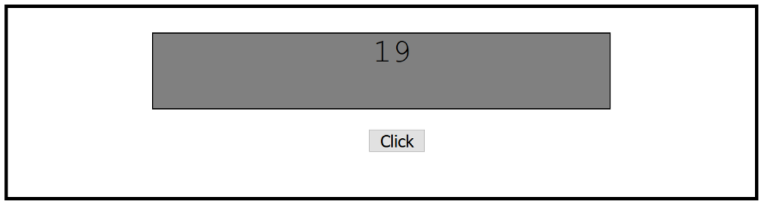
```
var i = 21;

function myFunction()
{
 i = i--; // the value of i will be remembered next function call
}
```

Each time your button is clicked, you should reduce the global variable by 1 and show the result. Your innerHTML of the paragraph with id “mydata” should show the current value of the global variable. When the variable gets to 0 or less, have the your innerHTML change to BOOM!

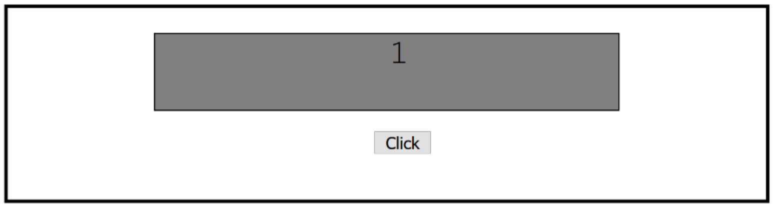
Your code should produce the exact results as the following figures. (i.e. it should not show 21, and it should show count down all the way to 0, and then show BOOM! for the following clicks on the button.)

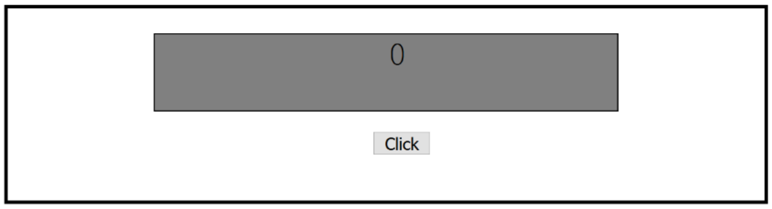

Count Down

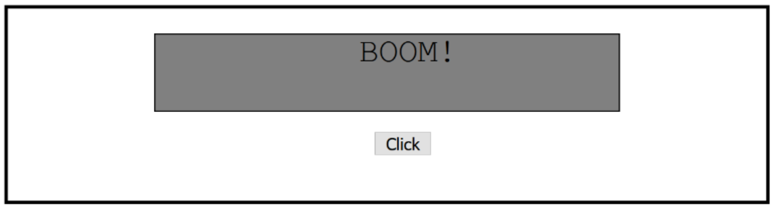




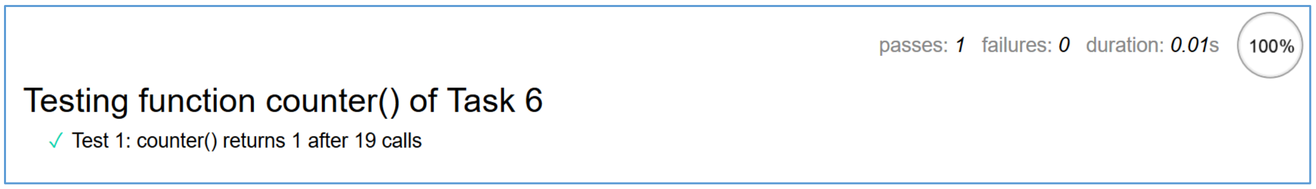
...







Once you have completed the step above, double click on `testRunner6.html`. You should see the following figure. If Test 1 fails, you must go to your `task6.js` and debug your code.

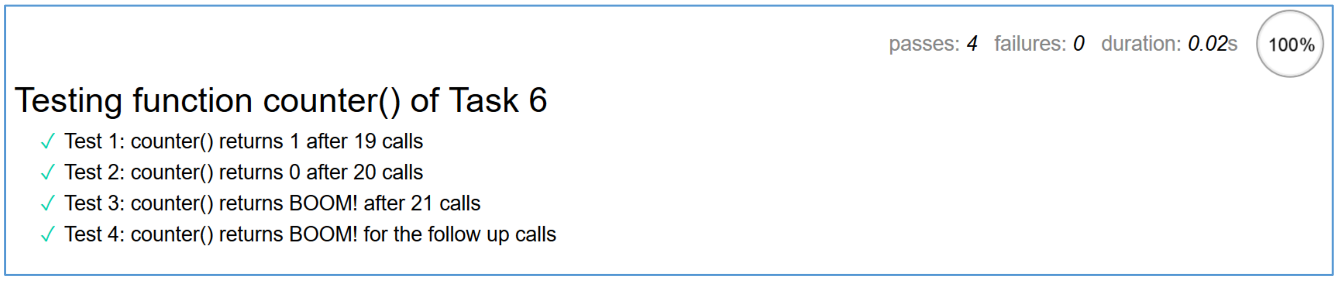


passes: 1 failures: 0 duration: 0.01s 100%

Testing function counter() of Task 6

✓ Test 1: counter() returns 1 after 19 calls

After you got the figure above, you may need to add 3 more test cases as explained in the `task6counterTest.js`. Once you complete it properly, you should see the following figure.



passes: 4 failures: 0 duration: 0.02s 100%

Testing function counter() of Task 6

✓ Test 1: counter() returns 1 after 19 calls
✓ Test 2: counter() returns 0 after 20 calls
✓ Test 3: counter() returns BOOM! after 21 calls
✓ Test 4: counter() returns BOOM! for the follow up calls

G. AFTER-LAB WORKS (THIS PART WILL NOT BE GRADED)

In order to review what you have learned in this lab as well as expanding your skills further, we recommend the following questions and extra practices:

1) Revisit objects `Math`, `Date`, and `document` in w3schools and explore more methods or properties of them by writing simple JavaScript codes.
   a. For object Math, learn more about `PI`, `round`, `pow`, `sqrt`, `abs`, `ceil`, `floor`, `sin`, `min`, `max`, etc. (https://www.w3schools.com/js/js_math.asp)
   b. For object `Date`, see 4 different constructors. (https://www.w3schools.com/js/js_dates.asp)
   c. For object `document`, see different methods such as `getElementsByTagName`, etc. (https://www.w3schools.com/js/js_htmldom_document.asp)
   d. Also see this fun animation made by html, css, JavaScript (https://www.w3schools.com/js/js_htmldom_animate.asp), and use your creativity to create some similar animations.
2) Revisit **arrays** and **strings** of JavaScript in w3schools. These two data structures have many applications in computer science and in your follow up courses. Interesting methods of strings include `indexOf()`, `lastIndexOf()`, `search()`, `slice()`, `substring()`, `substr()`, `replace()`, `concat()`, `trim()`, `charAT()`, `split()`, etc. More on strings here: https://www.w3schools.com/js/js_strings.asp
   More on arrays here: https://www.w3schools.com/js/js_arrays.asp
3) Add more algorithms and programs to your **Learning Kit** Project. You would need to have 40 buttons for 40 problems, flowcharts, and JavaScript solutions in your Learning kit and show it to your TA at the beginning of he lab next week (Lab 08).

Please feel free to discuss any of these questions in the course forum or see the TAs and/or Instructors for help.